UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/791,602	03/02/2004	Carl A. Waldspurger	A43	2029

69355   7590   08/05/2008
VMware, Inc.
DARRYL SMITH
3401 Hillview Ave.
PALO ALTO, CA 94304

EXAMINER
TURCHEN, JAMES R

ART UNIT	PAPER NUMBER
2139	

MAIL DATE	DELIVERY MODE
08/05/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

	Application No.	Applicant(s)
Office Action Summary	10/791,602	WALDSPURGER ET AL.
	Examiner	Art Unit
	JAMES TURCHEN	2139

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
  Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _22 April 2008_.
2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.
3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _2-4,6-19,21-33,35-44,46 and 47_ is/are pending in the application.
    4a) Of the above claim(s) _____ is/are withdrawn from consideration.
5)☐ Claim(s) _____ is/are allowed.
6)☒ Claim(s) _2-4,6-19,21-33,35-44,46 and 47_ is/are rejected.
7)☐ Claim(s) _____ is/are objected to.
8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.
10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.
    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
    a)☐ All b)☐ Some * c)☐ None of:
      1.☐ Certified copies of the priority documents have been received.
      2.☐ Certified copies of the priority documents have been received in Application No. _____.
      3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
    * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1)☐ Notice of References Cited (PTO-892)
2)☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3)☐ Information Disclosure Statement(s) (PTO/SB/08)
    Paper No(s)/Mail Date _____.

4)☐ Interview Summary (PTO-413)
    Paper No(s)/Mail Date. _____.
5)☐ Notice of Informal Patent Application
6)☐ Other: _____.

DETAILED ACTION

Claims 2-4, 6-19, 21-33, 35-44, 46, and 47 are pending. Claims 2-4, 6-9, 11-19, 21-33,

35-44, 46, and 47 are amended. Claims 1, 5, 20, 34, and 45 are cancelled. It is noted

that the word "memory" was left out of claim 5's cancelled matter. Also, (Currently

Amended) appears twice on claim 46. "A method as in claim 2" is also left out of claims

29 and 30 as being deleted matter.

Response to Arguments

Applicant's arguments with respect to claims 2-4, 6-19, 35-44, 46, and 47 have

been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly
claiming the subject matter which the applicant regards as his invention.

Claim 7 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite

for failing to particularly point out and distinctly claim the subject matter which applicant

regards as the invention. The term "potentially" is considered indefinite by examiner.

Claim Rejections - 35 USC § 103

Claims 2-4, 6-19, 21-28, 30-32, and 35-44 are rejected under 35 U.S.C. 103(a)

as being unpatentable over Nachenberg (US 5,826,013, hereinafter '013) in view of

Nachenberg (US 6,021,510, hereinafter '510).

Regarding claims 2 and 35:

013 discloses a method for verifying computer instructions in a computer that

includes at least one processor that executes instructions stored in a memory, the

memory being organized into separately addressable memory blocks, the method

comprising:

identifying a next instruction to be executed when executing a series of

instructions [figure 4A, 424, fetch instruction];

for the next instruction, determining an identifying value for a memory block that

contains the next instruction [figure 4B, 468, three bytes match?- 474, viral signature

match];

determining whether the identifying value satisfies a validation condition, wherein

the determining as to whether the identifying value satisfies the validation condition

requires comparing the identifying value of the memory block with a set of reference

values [*figure 4B, viral signature match is determined from a list of stored reference

values*];

allowing execution of the next instruction by the processor when the identifying

value satisfies the validation condition [figure 4B, 484, file is uninfected is the validation

condition; it is inherent to that a file is allowed to execute after being deemed virus-free];

and

whereby the next instruction is verified dynamically before being executed [*figure

4A, 438 proceed w/ emulation? fetches the next instruction*].

013 does not disclose generating a response when the identifying value does not

satisfy the validation condition. 510 discloses notifying the user when a virus is

detected [*column 4 lines 54-63*]. All the claimed elements were known in the prior art at

the time of invention and it would have been obvious to one of ordinary skill in the art to

modify the method of 013 with the notification to the user and checking to see if a file

has been modified [*figure 3*] as disclosed by 510 by known methods with no change in

their respective functions, and the combination would have yielded predictable results to

one of ordinary skill in the art at the time of invention.

Regarding claims 3 and 36:

013 and 510 disclose the method of claim 2, wherein the validation condition is

that the identifying value of the memory block matches any reference value in the set of

reference values [*510; column* 4 lines 48-53, the old hash value corresponds to the new

hash value].

Regarding claims 4 and 37:

013 and 510 disclose the method of claim 2, wherein the validation condition is

that the identifying value of the memory block differs from each reference value in the

set of reference values [*013; 474, viral signature match determines if the hash value of*

the instruction is different than all of the reference values].

Regarding claims 6, 39, and 40:

013 discloses a method for verifying computer instructions in a computer that

includes at least one processor that executes instructions stored in memory, the

memory being organized into separately addressable memory blocks, the method

comprising:

for at least one current instruction that has been identified for submission to the

processor for execution, computing a hash value as a function of a sub-set of contents

of a current memory block that contains the current instruction [*figure 4A, 424, fetch instruction; figure 4B, 468, three bytes match?- 474, viral signature match*];

determining whether the hash value satisfies a validation condition by comparing the hash value of the current memory block with a set of reference values [*figure 4B, viral signature match is determined from a list of stored reference values*];

if the hash value satisfies the validation condition, allowing execution of the current instruction by the processor [*figure 4B, 484, file is uninfected is the validation condition; it is inherent to that a file is allowed to execute after being deemed virus-free*];

wherein the computing of the hash value comprises applying a mask to the current memory block, the mask being a data structure that designates at least one byte of the current memory block to be ignored in the computing of the hash value, the data structure designating less than an entire memory block so that the hash value is based on only part of the contents of the current memory block [*column 12 lines 20-30, the selected page locations are scanned and the non-selected ones are ignored*].

013 does not disclose generating a response when the identifying value does not satisfy the validation condition. 510 discloses notifying the user when a virus is detected [*column 4 lines 54-63*]. All the claimed elements were known in the prior art at the time of invention and it would have been obvious to one of ordinary skill in the art to modify the method of 013 with the notification to the user and checking to see if a file has been modified [*figure 3*] as disclosed by 510 by known methods with no change in their respective functions, and the combination would have yielded predictable results to one of ordinary skill in the art at the time of invention.

Regarding claim 7:

013 and 510 disclose the method of claim 6, further comprising:

identifying, potentially non-constant contents of the current memory block, the non- constant contents being valid but changeable so that they do not indicate validity of the current memory block as a whole [*013, figure 5, checks for register modifications*]; and

configuring the mask so that the mask designates at least the non-constant contents to be ignored when generating the hash value [*013, figure 5, 512, exclude all viruses that cannot perform memory write with non-initialized index register*].

Regarding claims 8 and 41:

013 and 510 disclose the method of claim 2, further comprising:

for each of the separately addressable memory blocks, indicating in a structure whether the memory block is valid [*510, column 3 lines 46-54, register means includes hardware, software, and/or firmware registers, stacks, flags, automata, indication bits, etc.; a stack is a structure*];

accessing the structure to determine whether the memory block is valid prior to the determining of the identifying value [*013, column 11 lines 60-65, any affected portion is tagged*]; and

performing the determining of the identifying value when the structure does not indicate that the memory block is valid and directly allowing execution of the next instruction when the structure indicates that the memory block is valid [*013, column 12 lines 8-19, flagged items are scanned*].

Regarding claims 9 and 42:

013 and 510 disclose the method of claim 8, wherein the structure comprises a

group of hardware attribute indicators, and wherein the indicating in the structure

whether the plurality of memory blocks is validated comprises setting one of the

hardware attribute indicators, the one hardware attribute indicator corresponding to the

memory block [*510, column 3 lines 46-54, register means includes hardware, software,*

and/or firmware registers, stacks, flags, automata, indication bits, etc.].

Regarding claims 10 and 43:

013 and 510 disclose the method as in claim 9, in which the hardware attribute

indicators are execute and write permission attributes associated with an entry in a

translation lookaside buffer [*510, column 3 lines 46-54, register means includes*

hardware, software, and/or firmware registers, stacks, flags, automata, indication bits,

etc.].

Regarding claims 11 and 44:

013 and 510 disclose the method of claim 8, wherein the structure comprises a

software data structure, and wherein the indicating in the structure whether the plurality

of memory blocks is validated comprises making a corresponding entry in the software

data structure [*510, column 3 lines 46-54, register means includes hardware, software,*

and/or firmware registers, stacks, flags, automata, indication bits, etc.; a stack is a

structure].

Regarding claim 12:

013 and 510 disclose the method of claim 8, the determining of the identifying value for the memory block and the determining of whether the identifying value satisfies the validation condition are performed only when the structure does not indicate that the memory block is valid, the method further comprising:

modifying the structure so that the memory block is not indicated as being valid when the identifying value satisfies the validation condition [*510; column 3 line 58-column 4 line 64, the current sector matches a validation condition (matches the reference value) then it rescans the file for viruses; it is inherent to mark the file as virus-free or contaminated*].

Regarding claim 13:

013 and 510 disclose the method of claim 12, further comprising:

sensing modification of one of the memory blocks that the structure indicates is valid and, in response to the modification, setting its indication in the structure to indicate that the memory block is not valid [*013; figure 5, 502-512*].

Regarding claim 14:

013 and 510 disclose the method of claim 8, but does not disclose the method further comprising:

determining a branch history for the next instruction; and

checking whether the memory blocks in which instructions in the branch history are located are valid, the validation condition including the requirement that each checked memory block in the branch history is valid.

Examiner takes official notice that branch prediction is well known in the art at the time of invention. All the claimed elements were known and it would have been obvious to one or ordinary skill in the art to combine the elements by known methods with no change in their respective functions, and the combination would have yielded predictable results to one of ordinary skill in the art at the time of invention.

Regarding claim 15:

013 and 510 disclose the method of claim 2, wherein the determining of the identifying value and the determining as to whether the validation condition has been satisfied are performed only after a triggering event occurs [*it is inherent that a file be checked after the occurrence of a triggering event whether by a user, a time-based trigger, an active trigger, etc.*].

Regarding claim 16:

013 and 510 disclose the method of claim 15, wherein the triggering event is writing of at least one new unit of code or data to any physical component within the computer [*510, column 3 lines 35-36, the file changed thus causing a new unit of data*].

Regarding claim 17:

013 and 510 disclose the method of claim 15, but does not disclose in which the triggering event is an attempted execution of any instruction located on any unverified memory block. Examiner takes official notice that dynamically scanning a program as it is being opened or executed is known. It would have been obvious to one of ordinary skill in the art at the time of invention to modify the method of Nachenberg to scan a program in order to ensure the integrity of an executable program while it is running or

to detect infections between a programs integrity check and execution [A Generic Virus

Scanner in C++, page 6, section 2.5]

Regarding claim 18:

013 and 510 disclose the method of claim 15, wherein the triggering event is an

attempted execution of any instruction located on any unverified memory block of newly

installed software [*510, column 3 lines 26-40, the newly installed software would be

scanned on the fact that it is being examined for the first time*].

Regarding claim 19:

013 and 510 disclose the method of claim 15, further comprising triggering the

verification of the computer instructions depending on an identity of a user of the

computer, the user having caused the next instruction to be identified for execution [*it is

inherent that a computer OS has a method for access control based on the user of the

system in that the program will not load if the user does not have permission to run it*].

Regarding claim 21:

013 and 510 disclose the method of claim 15, further comprising triggering

dynamic verification depending on a context in which the next instruction is submitted

for execution, wherein the context is a level of security clearance associated with the

computer, a user of the computer, or a program of which the next instruction is a part [*it

is inherent that a computer OS has a method for access control based on the user of

the system in that the program will not load if the user does not have permission to run

it*].

Regarding claim 22:

013 and 510 disclose the method of claim 2, wherein the identifying of the next

instruction is performed for only a sample of the series of instructions [*510, figure 1*

shows the file is divided into a plurality of sectors and only sectors 1,2,3 and J are

scanned].

Regarding claim 23:

013 and 510 disclose the method of claim 22, wherein the sample is a time-

sampled sub-set of the series of instructions [*it is inherent that the virus scanner*

disclosed by Nachenberg relies upon some period of time elapsing between samples].

Regarding claim 24:

013 and 510 disclose the method of claim 22, wherein the sample is a

sequentially sampled sub-set of the series of instructions [*510, figure 1 shows the*

sectors 1, 2 and 3; it is inherent that these sectors would be sequentially sampled].

Regarding claim 25:

013 and 510 disclose the method of claim 22, wherein the sample is a sub-set of

the series of instructions sampled spatially, the sampling being over a range of memory

block identifiers [*510, figure 1 shows the sectors 1, 2, 3 are a set size and spatially*

sampled].

Regarding claim 26:

013 and 510 disclose the method of claim 2, but does not disclose wherein the

response comprises termination of a software entity with which the current memory

block is associated. Examiner takes official notice that suspending or cancelling a

current software's execution when a virus has been detected was well known in the art

at the time of invention. All the claimed elements were known in the prior art and it

would have been obvious to one skilled in the art to combine the elements as claimed

by known methods with no change in their respective functions, and the combination

would have yielded predictable results to one of ordinary skill in the art at the time of

invention.

Regarding claim 27:

013 and 510 disclose the method of claim 2, wherein the response comprises

suspension of execution of a software entity with which the current memory block is

associated. Examiner takes official notice that suspending or cancelling a current

software's execution when a virus has been detected was well known in the art at the

time of invention. All the claimed elements were known in the prior art and it would

have been obvious to one skilled in the art to combine the elements as claimed by

known methods with no change in their respective functions, and the combination would

have yielded predictable results to one of ordinary skill in the art at the time of invention.

Regarding claim 28:

013 and 510 disclose the method of claim 2, wherein the response comprises a

message posted to a user, system administrator, or other predetermined recipient [*510,

column 4 lines 48-63, sends a message to the user to via the interface*].

Regarding claim 30:

013 and 510 disclose the method of claim 2, wherein: the computer includes a

virtual machine running on an underlying hardware platform via an intermediate

software layer; and the response includes checkpointing the state of the virtual machine

[*013, column 3 line 6-18, the CPU emulator is a virtual machine that tricks the virus into thinking it is being run on the CPU*].

Regarding claim 31:

013 and 510 disclose the method of claim 2, wherein the response is a first possible response, the method further comprising: associating the first possible response with the memory block; associating a second possible response with a different memory block; upon detection of failure of the next instruction to satisfy the validation condition, identifying which one of the possible responses is associated with the memory block, and generating the one possible response associated with the memory block in which the next instruction is located [*510, column 4 lines, determination is sent to the user, it is inherent that the determination will be different for different memory*].

Regarding claim 32:

013 and 510 disclose the method of claim 2, further comprising: associating reference values from the set of reference values with respective programs such that each association signifies that the reference value corresponds to a memory block storing instructions for one of the programs; and tracking which of the respective programs is being executed and the association between the matching reference value and the corresponding one of the programs [*510, column 1 lines 27-36, a hash value is associated with a program/file*].

Claims 29, 33, 46, and 47 are rejected under 35 U.S.C. 103(a) as being

unpatentable over 013 and 510 as applied to claims 2 and 35 above, and further in view

of SimOS.

013 and 510 disclose the method of claim 2, wherein:

the computer includes a virtual machine running in a direct execution mode on an

underlying hardware platform via an intermediate software layer [*013, column 3 lines 6-

16, the CPU emulator is a virtual machine that tricks the virus into thinking it is being run

on the CPU*], but does not disclose the response comprises a switching of an execution

mode of the virtual machine from the direct execution mode to a binary translation

mode. SimOS discloses the ability to switch between direct execution and binary

translation modes [*page 40, Switching simulators and sampling*]. It would have been

obvious to one or ordinary skill in the art at the time of invention to modify the method of

013 and 510 to allow switching of modes in order to dynamically generate code

supports on-the-fly changes of the simulator's level of detail [*page 39, column 2

paragraph 3*].

Conclusion

Applicant's amendment necessitated the new ground(s) of rejection presented in

this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP

§ 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37

CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE

MONTHS from the mailing date of this action. In the event a first reply is filed within

TWO MONTHS of the mailing date of this final action and the advisory action is not

mailed until after the end of the THREE-MONTH shortened statutory period, then the

shortened statutory period will expire on the date the advisory action is mailed, and any

extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of

the advisory action. In no event, however, will the statutory period for reply expire later

than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the

examiner should be directed to JAMES TURCHEN whose telephone number is

(571)270-1378. The examiner can normally be reached on MTWRF 7:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's

supervisor, Kristine Kincaid can be reached on (571)272-4063. The fax phone number

for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the

Patent Application Information Retrieval (PAIR) system. Status information for

published applications may be obtained from either Private PAIR or Public PAIR.

Status information for unpublished applications is available through Private PAIR only.

For more information about the PAIR system, see http://pair-direct.uspto.gov. Should

you have questions on access to the Private PAIR system, contact the Electronic

Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a

USPTO Customer Service Representative or access to the automated information

system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

JRT

/Kristine Kincaid/
Supervisory Patent Examiner, Art Unit 2139